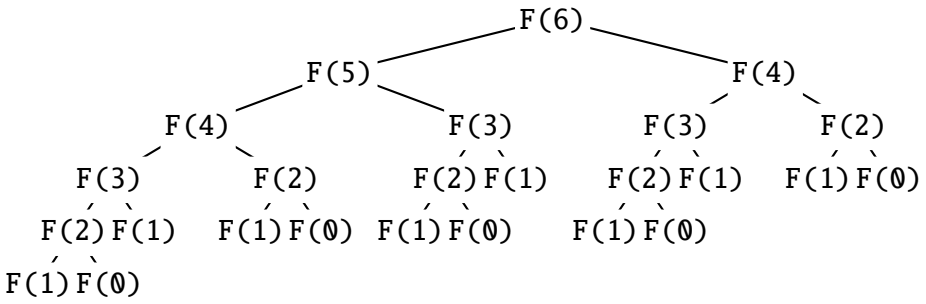
Lycée Les 3 Sources **Bourg-Lès-Valence**
NSI T^le – Thème 5 Année **2025-26**

Programmation dynamique

Diviser pour régner, mais en mieux

Comme nous l'avons déjà vu, l'approche *diviser pour régner* consiste à découper un problème en sous-problèmes plus simples à résoudre, jusqu'à arriver à des problèmes triviaux. La solution finale est alors obtenue à partir des solutions des sous-problèmes. Cette approche n'est pas forcément la plus efficace, notamment en temps de calculs.
Si on prend l'approche diviser pour régner pour calculer un terme de la suite de Fibonacci, on obtient l'arbre d'appels ci-dessous. On peut remarquer que beaucoup d'appels sont répétés. Par exemple, `F(2)` est calculé 5 fois. Si on augmente la valeur de `n`, le temps de calcul de `F(n)` va rapidement exploser.



La **programmation dynamique** est une amélioration de la méthode diviser pour régner. La différence, c'est que les sous-problèmes, s'ils se répètent, ne sont résolus qu'une seule fois. L'idée consiste à partir des cas de base et à calculer les valeurs intermédiaires, jusqu'à arriver au problème considéré. Les valeurs intermédiaires sont souvent stockées dans un tableau, un dictionnaire ou toute autre structure adaptée.
Dans le cas de la suite de Fibonacci, cela revient à partir de `F(0)`, `F(1)`, puis calculer tous les termes successifs jusqu'à arriver à `F(n)`. La complexité est alors linéaire.

Le retour du problème du sac à dos

L'année dernière, nous avons étudié le problème du sac à dos. Étant donné une liste de n objets, chacun ayant un poids et une valeur, il faut déterminer la valeur maximale que l'on peut mettre dans un sac à dos pouvant supporter un poids limité. La méthode exhaustive consiste à tester toutes les combinaisons. La complexité est $O(2^n)$. Avec un algorithme glouton, il est possible de trouver une valeur approchée avec une complexité en $O(n \log n)$. Il faut trier les objets par rentabilité (valeur/poids) et ensuite prendre systématiquement l'objet le plus rentable et avec la plus grande valeur qui peut rentrer dans le sac.

Exercice 1 : Nous allons considérer les objets ci-contre.

1) Calculer la rentabilité de chaque objet.
2) Classer les objets par ordre décroissant de rentabilité.
3) Déterminer la valeur maximale que peut contenir un sac à dos pouvant supporter 15 kg maximum, d'après l'algorithme glouton.

Obj.	kg	€	€/kg	Ordre
1	2	4		
2	3	2		
3	1	5		
4	8	12		
5	2	6		
6	3	7		

La solution trouvée avec l'algorithme glouton n'est pas forcément la meilleure. Elle a juste été trouvée de façon rapide. Mais une fois qu'un choix a été fait, il n'est jamais remis en

question. Avec l'approche dynamique, on va chercher les solutions optimales en prenant en compte les objets un par un. Dans le tableau ci-dessous, les 3 premières lignes sont remplies. Si on n'a pas d'objets disponibles, on n'a que la valeur 0. Avec le premier objet, on peut avoir 4 à partir de 2 kg. Avec le deuxième objet, en dessous de 5 kg, il fait garder la solution avec le pemier objet et ensuite, on peut prendre les deux, et donc avoir un total de 6.
De façon générale, pour remplir la case de l'objet `i`, avec un maximum de `j` kg, il faut regarder s'il vaut mieux garder la solution trouvée avec les objets précédents, ou si on peut prendre l'objet et ajouter sa valeur à la solution optimale avec les objets précédents avec la capacité restante.
Si on note `tab[i][j]` la solution optimale avec les `i` premiers objets et une limite de `j` kg, ainsi que `val(i)` la valeur de l'objet `i` et `poids(i)` son poids, alors :

$$\texttt{tab[i][j]} = \begin{cases} \texttt{tab[i-1][j]} & \text{si } \texttt{poids(i)} > \texttt{j} \\ \texttt{max(tab[i-1][j], val(i) + tab[i-1][j-poids(i)])} & \text{sinon} \end{cases}$$

Objets			Limite de poids, en kg															
Nom	kg	€	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	4	0	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	3	2	0	0	4	4	4	6	6	6	6	6	6	6	6	6	6	6
3	1	5	0	5														
4	8	12	0															
5	2	6	0															
6	3	7	0															

Exercice 2 : On considère le tableau ci-dessus.

1) L'objet 3 pèse 1 kg et vaut 5 €. On considère la limite de 2 kg.
   a) Quelle est la valeur maximale qui peut être prise avec une limite de 2 kg avec uniquement les 2 premiers objets ?
   b) Si on prend l'objet 3, quelle est la limite de poids qui reste pour prendre les deux premiers objets ?
   c) Quelle est donc la valeur maximale qui peut être prise avec une limite de 2 kg et les 3 premiers objets ?
2) Toujours avec l'objet 3, déterminer la valeur maximale qui peut être obtenue avec une limite de 3 kg.
3) Compléter le reste du tableau et déterminer la solution optimale à ce problème.

Programmation dynamique et mémoïsation

Parfois, il n'est pas évident de déterminer dans quel ordre résoudre les sous-problèmes pour arriver facilement au résultat final. Par exemple, comment déterminer l'altitude maximale à partir d'un nombre donné pour la suite de Syracuse ? Connaître l'altitude maximale pour les nombres plus petits ne permet pas forcément de trouver facilement celle du suivant.
L'idée dans un cas comme cela est de reprendre l'approche diviser pour régner et de rajouter un dictionnaire pour stocker les résultats intermédiaires. Cette technique s'appelle la **mémoïsation**. On garde le principe de l'appel récursif, mais on le modifie de la manière suivante :

- Si la solution cherchée est dans le dictionnaire, on la renvoie, évitant ainsi les calculs intermédiaires.
- Sinon, on la calcule, on la stocke dans le dictionnaire et on la renvoie.

L'avantage c'est qu'on allie la simplicité de la méthode diviser pour régner et l'efficacité de la programmation dynamique. Par contre le coût en mémoire peut être plus important qu'avec une vraie approche par programmation dynamique.